

# 计算机组成

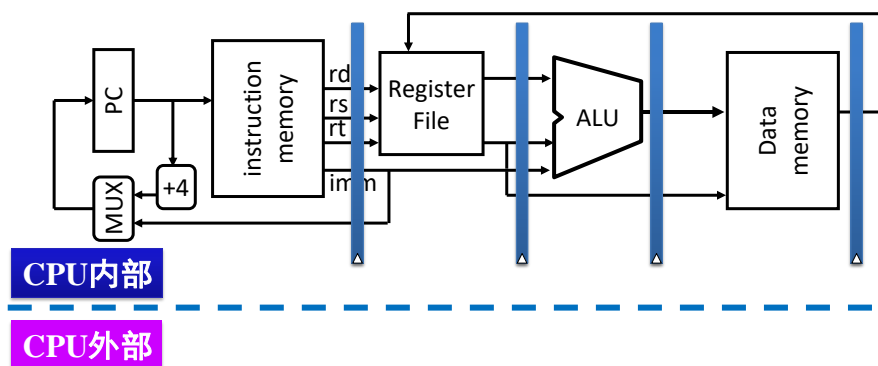
## 支持I/O

高小鹏

北京航空航天大学计算机学院

### 数据通路：增加信号

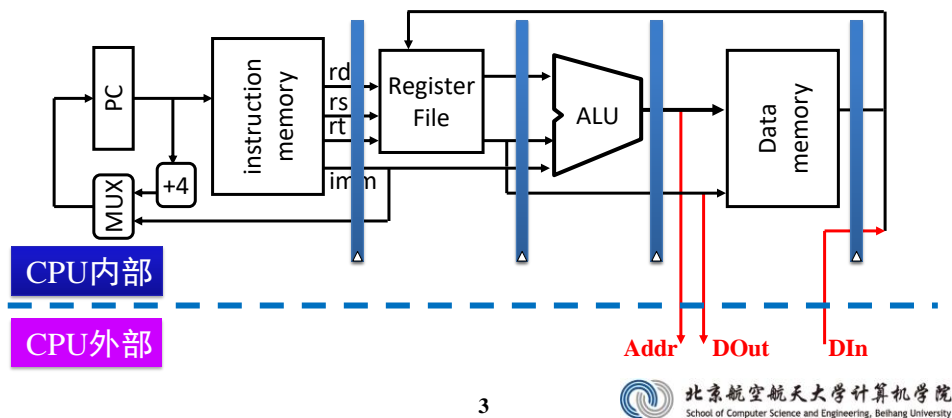
- 增加必要的地址、数据



## 数据通路：增加信号

### 增加必要的地址、数据

- ◆ Addr: ALU计算的存储器地址
- ◆ DOut: CPU写数据
- ◆ DIn: CPU读入的数据



3



北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

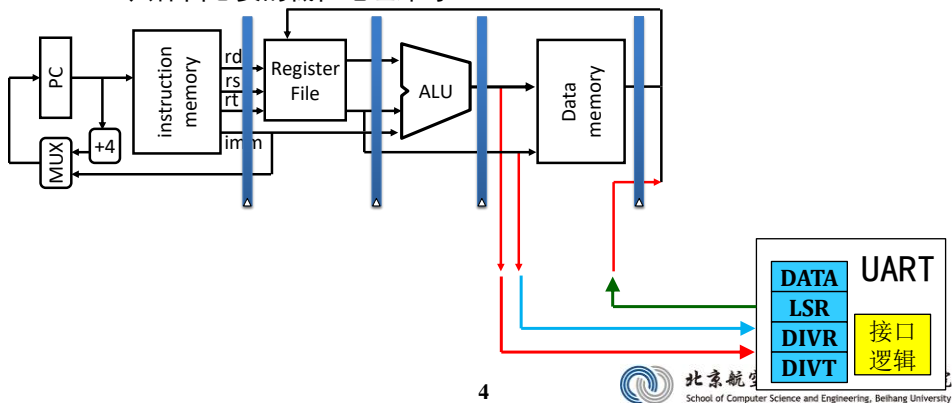
## 支持对I/O的访问：与设备对接

### 每个设备都有自己的 $addr_{dev}$ 、 $din$ 、 $dout$

- ◆  $Addr_{dev}$ : 选择设备内部的寄存器。其本质是offset
- ◆ 设备内部的寄存器数量少，因此 $Addr_{dev}$ 位数少

### Q: $Addr_{CPU}$ 位数多，怎么处理？

- ◆ A: 只保留必要的低位地址即可



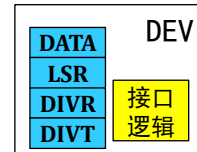
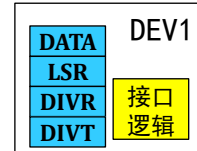
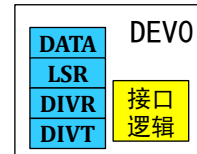
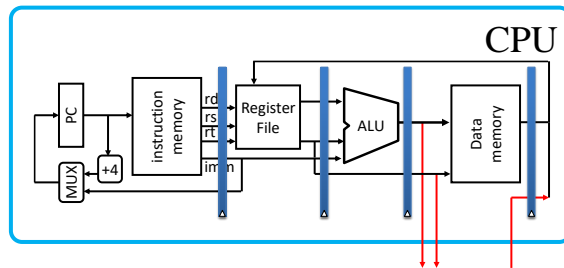
4



北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## 多个设备怎么办？

- CPU不能为每个设备都提供一套地址/数据
  - 否则会导致CPU设计变得复杂

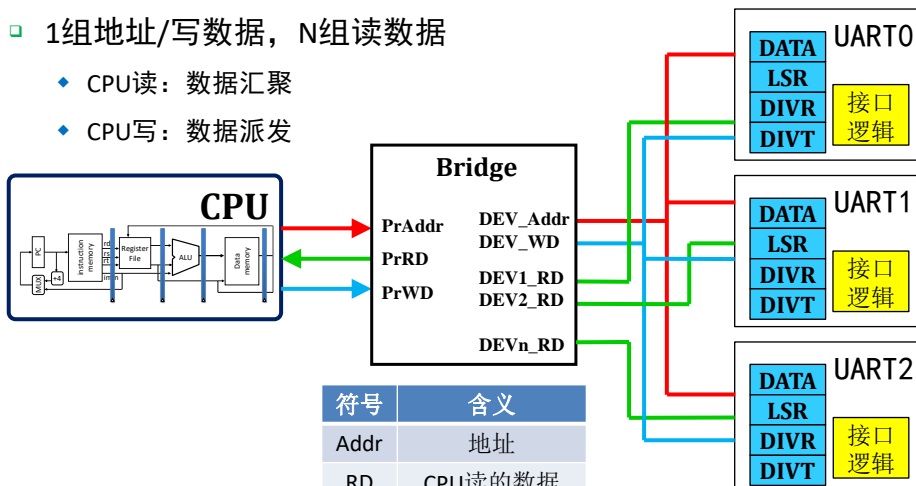


5

计算机学院

## 增加新模块：Bridge

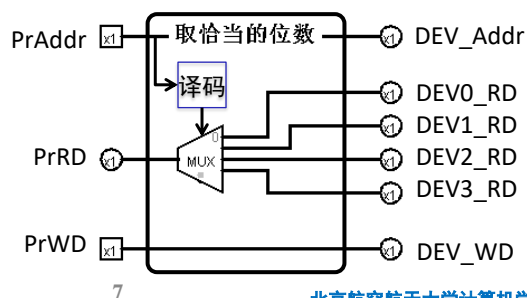
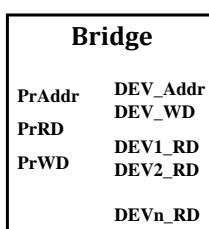
- Bridge：类似与网络switch
  - ◆ CPU侧：1组接口。设备侧：N组接口
- 1组地址/写数据，N组读数据
  - ◆ CPU读：数据汇聚
  - ◆ CPU写：数据派发



## Bridge功能及内部结构

- 完成地址、数据转换，控制信号的产生

- 地址
- 读数据
- 写数据



北京航空航天大学计算机学院

## 地址图

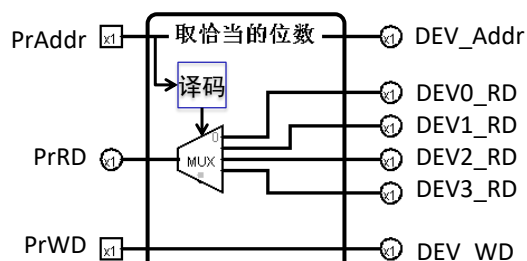
- 地址图：所有设备在地址空间的分布区域

- CPU读写设备(其实是程序员)必须知道设备地址
- Bridge也必须知道设备，否则无法完成译码

- 示例

- 设备0~2均需要256B的地址空间
- 设备3需要1MB的地址空间

设备	MIPS地址范围	占用空间
DEV0	A0000000 <sub>H</sub> ~ A00000FF <sub>H</sub>	256字节
DEV1	A0000100 <sub>H</sub> ~ A00001FF <sub>H</sub>	256字节
DEV2	A0000200 <sub>H</sub> ~ A00002FF <sub>H</sub>	256字节
DEV3	A0100000 <sub>H</sub> ~ A01FFFFF <sub>H</sub>	1MB字节



北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## Bridge如何理解CPU的32地址？

□ 假设CPU要读写DEV0，则PrAddr必然为A00000XX。分析如下：

- ◆ 1) 只要PrAddr<sub>31..08</sub>=A00000，则CPU必然读写DEV0
- ◆ 2) 只要PrAddr<sub>31..08</sub>≠A00000，则CPU必然不读写DEV0
- ◆ 3) 显然，PrAddr<sub>07..00</sub>只决定访问DEV0内部的哪个寄存器

□ 同理对于DEV3可得如下结论

- ◆ 1) 只要PrAddr<sub>31..20</sub>=A01，则CPU必然读写DEV3
- ◆ 2) 只要PrAddr<sub>31..20</sub>≠A01，则CPU必然不读写DEV3
- ◆ 3) PrAddr<sub>19..00</sub>决定访问DEV3内部的哪个寄存器

设备	MIPS地址范围	占用空间
DEV0	A0000000 <sub>H</sub> ~ A00000FF <sub>H</sub>	256字节
DEV1	A0000100 <sub>H</sub> ~ A00001FF <sub>H</sub>	256字节
DEV2	A0000200 <sub>H</sub> ~ A00002FF <sub>H</sub>	256字节
DEV3	A0100000 <sub>H</sub> ~ A01FFFFF <sub>H</sub>	1MB字节

9



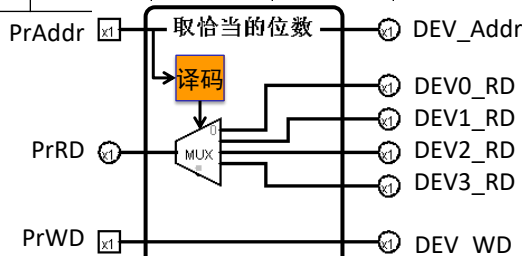
北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## Bridge如何理解CPU的32地址？

□ 对于设备，其基地址位数/值均是固定的，偏移位数也是固定的

设备	MIPS地址范围	占用空间	基地址	偏移
DEV0	A0000000 <sub>H</sub> ~ A00000FF <sub>H</sub>	256字节	A00000	8位
DEV1	A0000100 <sub>H</sub> ~ A00001FF <sub>H</sub>	256字节	A00001	8位
DEV2	A0000200 <sub>H</sub> ~ A00002FF <sub>H</sub>	256字节	A00002	8位
DEV3	A0100000 <sub>H</sub> ~ A01FFFFF <sub>H</sub>	1MB字节	A01	20位

□ 本质是设备基地址译码



10



北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## Bridge如何知道CPU当前在读写哪个设备？

- 为了判断CPU地址是否读写某个设备，只需要将CPU地址的高位部分与该设备的基地址进行比较，产生一个HIT
- 例如对于DEV0，只需要判断PrAddr<sub>31..08</sub>是否为A00000即可

```
assign HitDEV0 = (PrAddr[31:08] == 20'hA00000) ;
assign HitDEV1 = (PrAddr[31:08] == 20'hA00001) ;
assign HitDEV2 = (PrAddr[31:08] == 20'hA00002) ;
assign HitDEV3 = (PrAddr[31:20] == 12'hA01) ;
```

设备	MIPS地址范围	占用空间	基地址	偏移
DEV0	A00000 <sub>00</sub> <sub>H</sub> ~ A00000 <sub>FF</sub> <sub>H</sub>	256字节	A00000	8位
DEV1	A00001 <sub>00</sub> <sub>H</sub> ~ A00001 <sub>FF</sub> <sub>H</sub>	256字节	A00001	8位
DEV2	A00002 <sub>00</sub> <sub>H</sub> ~ A00002 <sub>FF</sub> <sub>H</sub>	256字节	A00002	8位
DEV3	A0100000 <sub>00</sub> <sub>H</sub> ~ A01FFFFF <sub>FF</sub> <sub>H</sub>	1MB字节	A01	20位

11



北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## Bridge如何给设备输出地址？

- DEV0，需要CPU地址的最低8位偏移，即PrAddr<sub>07..00</sub>
- DEV1，需要CPU地址的最低8位偏移，即PrAddr<sub>07..00</sub>
- DEV2，需要CPU地址的最低8位偏移，即PrAddr<sub>07..00</sub>
- DEV3，需要CPU地址的最低20位偏移，即PrAddr<sub>19..00</sub>
- 因此，bridge只需输出PrAddr<sub>19..00</sub>即可，而DEV0-DEV3各取所需

设备	MIPS地址范围	占用空间	基地址	偏移
DEV0	A00000 <sub>00</sub> <sub>H</sub> ~ A00000 <sub>FF</sub> <sub>H</sub>	256字节	A00000	8位
DEV1	A00001 <sub>00</sub> <sub>H</sub> ~ A00001 <sub>FF</sub> <sub>H</sub>	256字节	A00001	8位
DEV2	A00002 <sub>00</sub> <sub>H</sub> ~ A00002 <sub>FF</sub> <sub>H</sub>	256字节	A00002	8位
DEV3	A0100000 <sub>00</sub> <sub>H</sub> ~ A01FFFFF <sub>FF</sub> <sub>H</sub>	1MB字节	A01	20位

12

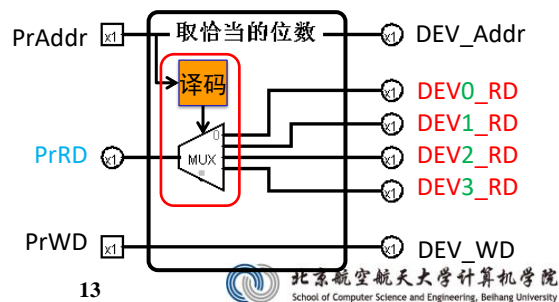


北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## Bridge如何支持CPU读设备？

- 每个设备都在向bridge输出各自的DEV?\_RD
  - 无论CPU是否访问自己，每个设备都会根据输入的地址偏移“自做多情”的输出数据

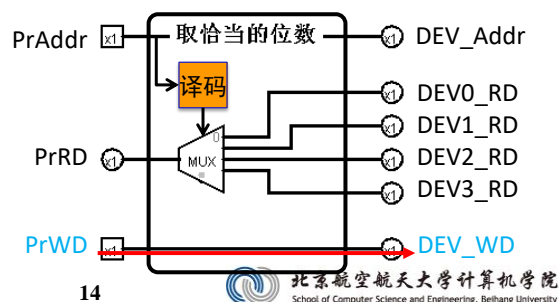
```
assign PrRD = HitDEV0 ? DEV0_RD :
              HitDEV1 ? DEV1_RD :
              . . .
              DEV3_RD ;
```



## Bridge如何支持CPU写设备？

- CPU写数据：连接至所有设备的数据输入
  - 直通输出，不需要再转换
- 每个设备都必须有一个独立的写使能We
  - 满足2个条件：①CPU写使能有效，②命中设备
- 以DEV3的WeDEV3为例

```
assign WeDEV3 = WeCPU & HitDEV3 ;
```



## 防止误写DM

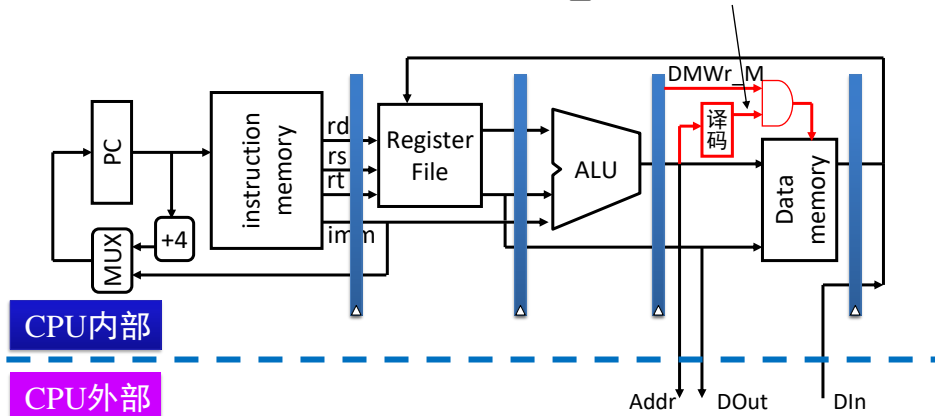
Q: HitDM逻辑部署在哪里?

A: 只能部署在M级

问题: store指令可能写DM, 也可能写设备

方案: DM的写入信号需要增加对DM地址范围的判断信息

DM的实际写入信号 =  $DMWr\_M \& HitDM$



15

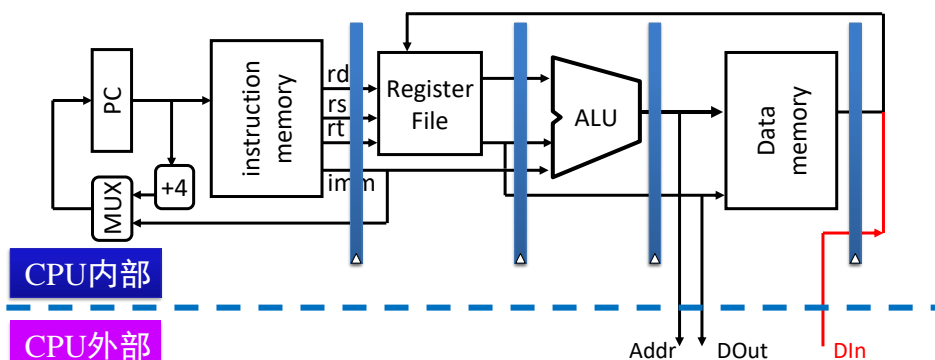


北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## 支持读I/O寄存器<sup>1/2</sup>

无I/O时, RF回写数据来自{DM、ALU、PC4}, 其MUX控制信号

MUXWD\_Sel = load指令 ? DR@W :  
cal指令 ? AO@W :  
PC4@W



16



北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University



## 支持读I/O寄存器<sup>2/2</sup>

- 无I/O时，RF回写数据来自{DM、ALU、PC4}，其MUX控制信号

$$\text{MUXWD\_Sel} = \begin{matrix} \text{load指令} & \& \text{HitDM} & ? & \text{DR@W} & : \\ \text{load指令} & \& \text{!HitDM} & ? & \text{DIn@W} & : \\ \text{cal指令} & & & ? & \text{AO@W} & : \\ & & & & \text{PC4@W} & : \end{matrix}$$

